Using High Performance Computing to Realize a System-Level RBDO for Military Ground Vehicles

- **David A. Lamb, Ph.D.**
- Computational Reliability and Safety Research team

RDECOM

TARDEC

U.S. ARMY TANK AUTOMOTIVE RESEARCH, DEVELOPMENT AND ENGINEERING CENTER

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **14 JUL 2008**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE
Using High Performance Computing to Realize a System-Level RDDO for Military Ground Vehicles

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
David A. Lamb, Ph.D

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army RDECOM-TARDEC 6501 E 11 Mile Rd Warren, MI 48397-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
18951

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)
TACOM/TARDEC

11. SPONSOR/MONITOR’S REPORT NUMBER(S)
18951

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at theDepartment of Defense (DoD) High Performance Computing (HPC) Users Group Conference" which was held July 14-17, 2008, in Seattle, Washington, The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**20**	

The Team

The University of Iowa

University of Maryland

GP Technologies, Inc.

TARDEC:

- David A. Lamb, Ph.D.
- Dr. David Gorsich

University of Iowa:

- Prof. K.K. Choi
- Dr. Ed Hardee

University of Maryland:

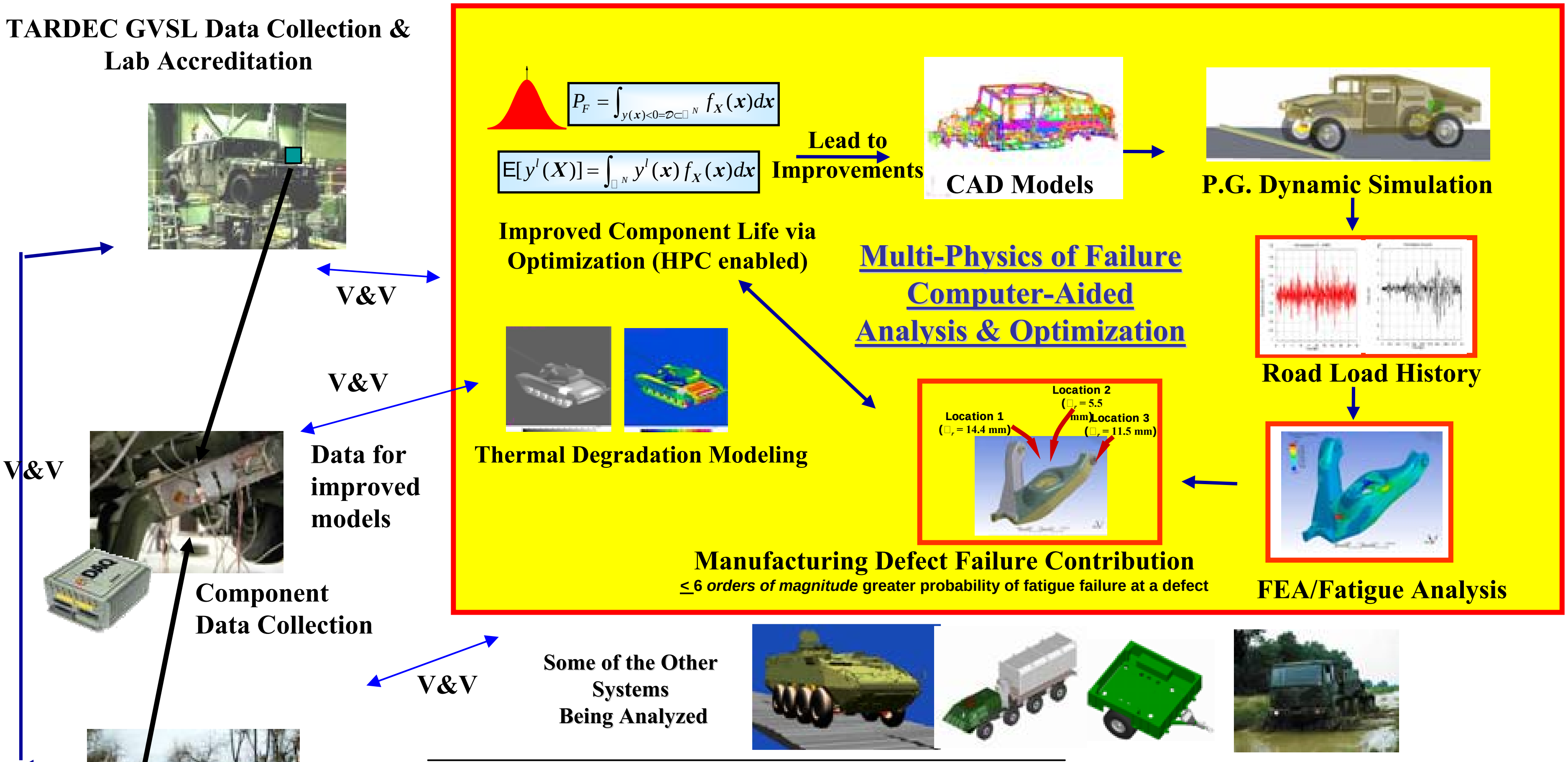
- Prof. B.D. Youn

Ghiocel Predictive Technologies:

- Dr. Dan Ghiocel

AND OTHERS

High Level View of TARDEC Reliability Modeling Efforts



TARDEC GVSL Data Collection & Lab Accreditation

V&V

V&V

Data for improved models

V&V

Component Data Collection

Field Data Collection and Analysis

V&V

$$P_F = \int_{y(x)<0=\mathcal{D}\subset\square^N} f_X(x)dx$$

$$\mathrm{E}[y^l(X)] = \int_{\square^N} y^l(x) f_X(x)dx$$

Lead to Improvements

CAD Models

P.G. Dynamic Simulation

Improved Component Life via Optimization (HPC enabled)

Multi-Physics of Failure Computer-Aided Analysis & Optimization

Road Load History

Thermal Degradation Modeling

Location 1 ($\square_r$ = 14.4 mm)

Location 2 ($\square_r$ = 5.5 mm)

Location 3 ($\square_r$ = 11.5 mm)

FEA/Fatigue Analysis

Manufacturing Defect Failure Contribution

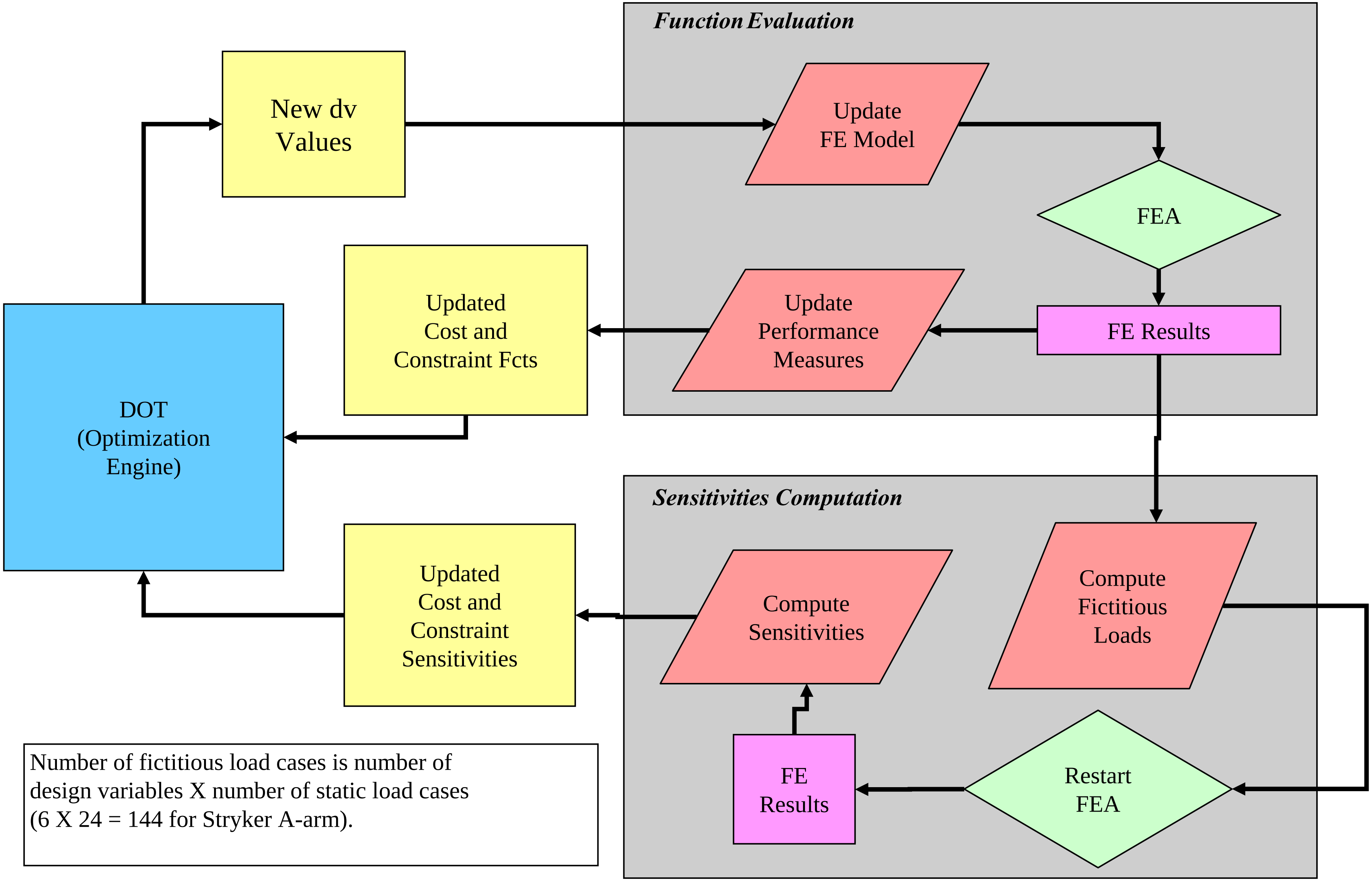
< 6 *orders of magnitude* greater probability of fatigue failure at a defect

Some of the Other Systems Being Analyzed

Technology Developments being Leveraged from Auto Industry & ARC

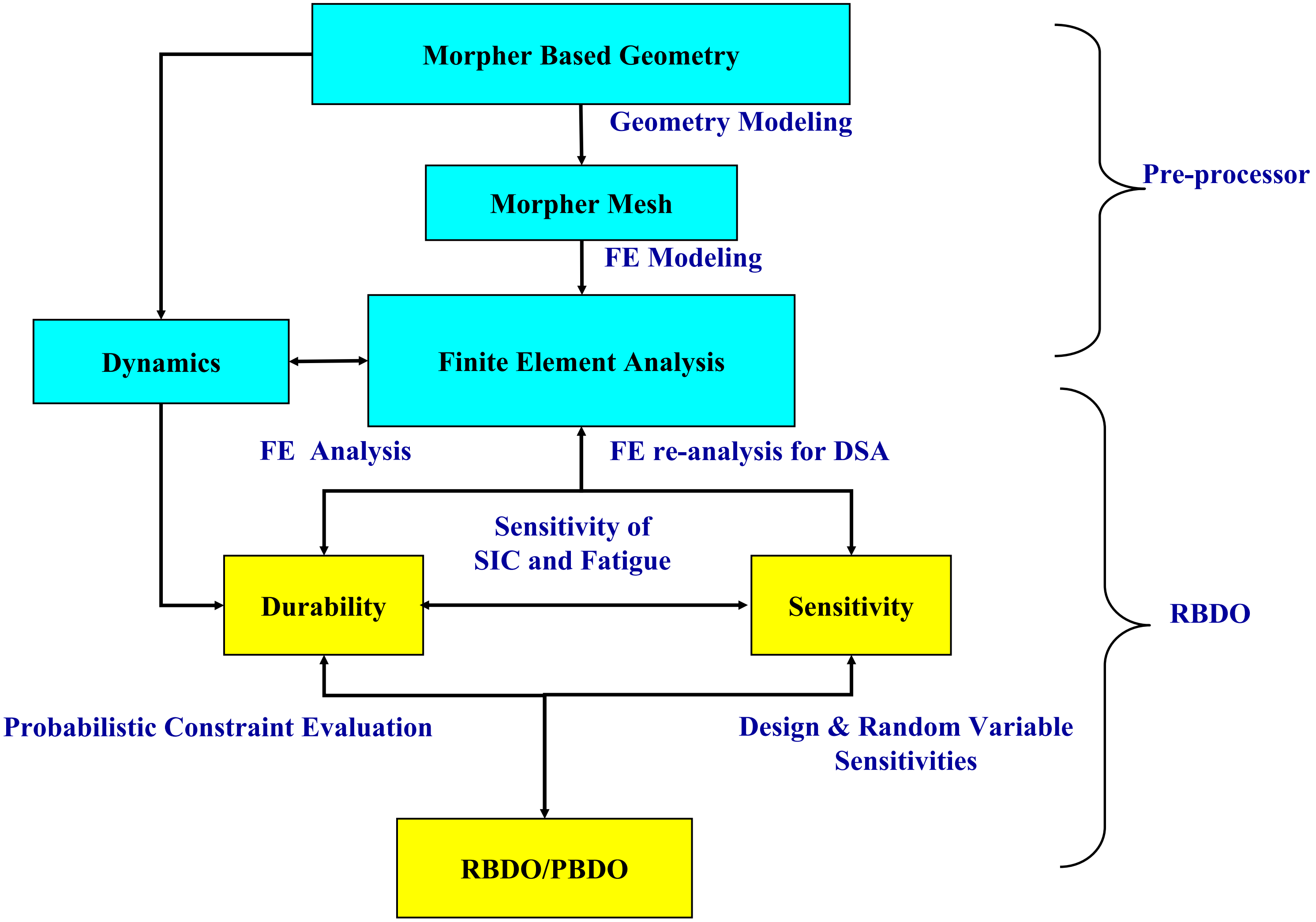
Ford nCode TOYOTA GM IOWA AMSAA VEXTEC DAIMLERCHRYSLER MSC SOFTWARE. LMS SAE INTERNATIONAL

The Reliability Algorithm


Function Evaluation
New dv Values
Update FE Model
FEA
FE Results
Update Performance Measures
Updated Cost and Constraint Fcts
DOT (Optimization Engine)
Sensitivities Computation
Compute Fictitious Loads
Restart FEA
FE Results
Compute Sensitivities
Updated Cost and Constraint Sensitivities

Number of fictitious load cases is number of design variables X number of static load cases (6 X 24 = 144 for Stryker A-arm).

RBDO Flowchart



Morpher Based Geometry

Geometry Modeling

Morpher Mesh

FE Modeling

Pre-processor

Dynamics

Finite Element Analysis

FE Analysis

FE re-analysis for DSA

Sensitivity of
SIC and Fatigue

Durability

Sensitivity

RBDO

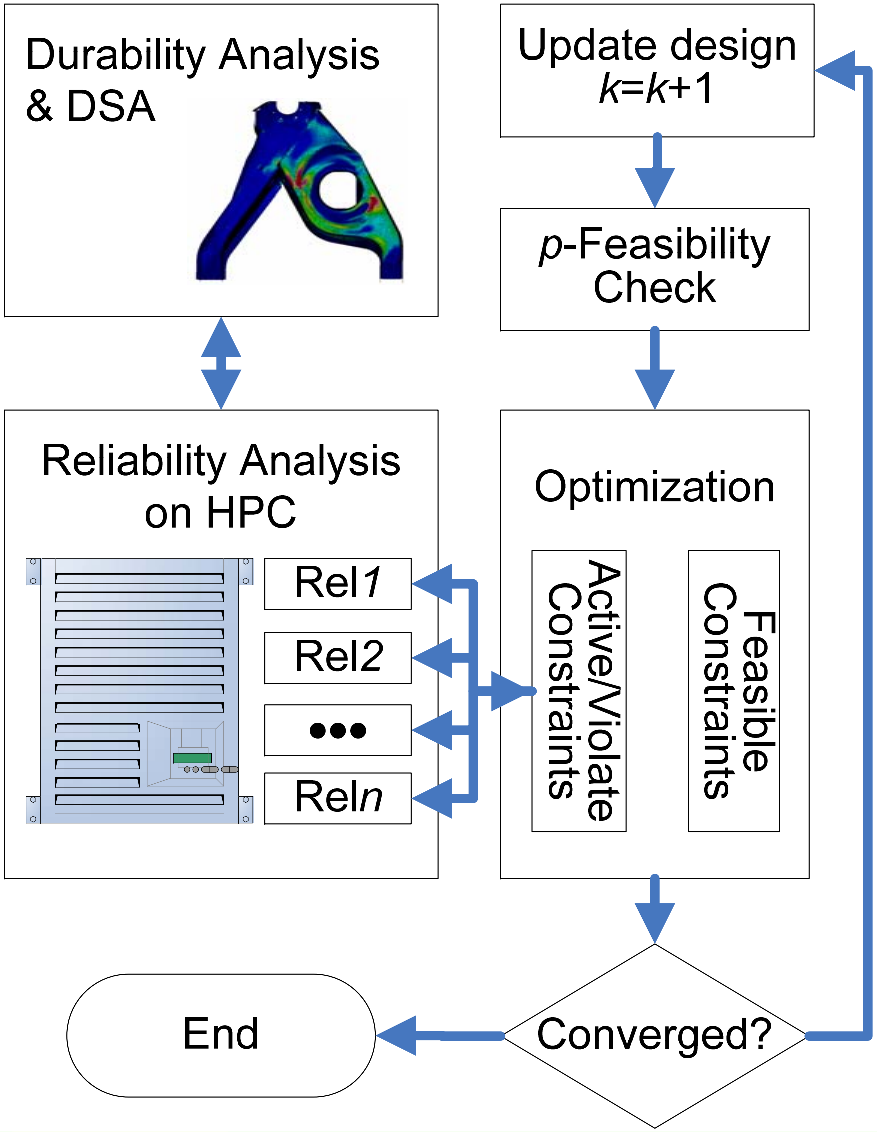
Probabilistic Constraint Evaluation

Design & Random Variable
Sensitivities

RBDO/PBDO

🟨 Reliability/Fatigue Analysis Software

Optimization Loop



Durability Analysis & DSA

Update design $k=k+1$

p-Feasibility Check

Reliability Analysis on HPC

Rel*1*

Rel2

●●●

Rel*n*

Optimization

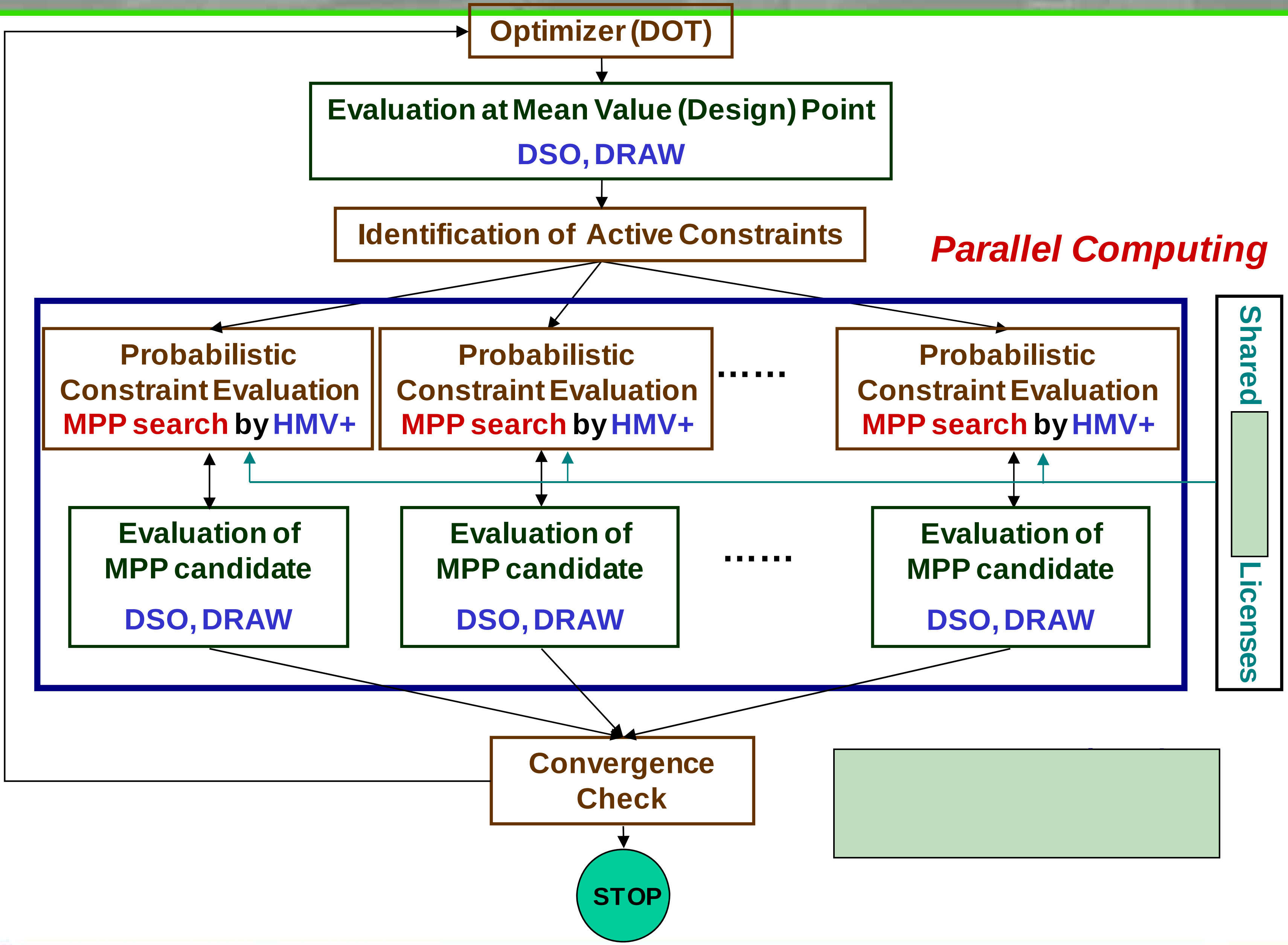
Active/Violate Constraints

Feasible Constraints

Converged?

End

Parallel Computing for RBDO using HPC



Optimizer (DOT)

Evaluation at Mean Value (Design) Point

DSO, DRAW

Identification of Active Constraints

Parallel Computing

Probabilistic Constraint Evaluation MPP search by HMV+

Probabilistic Constraint Evaluation MPP search by HMV+

……

Probabilistic Constraint Evaluation MPP search by HMV+

Shared Licenses

Evaluation of MPP candidate DSO, DRAW

Evaluation of MPP candidate DSO, DRAW

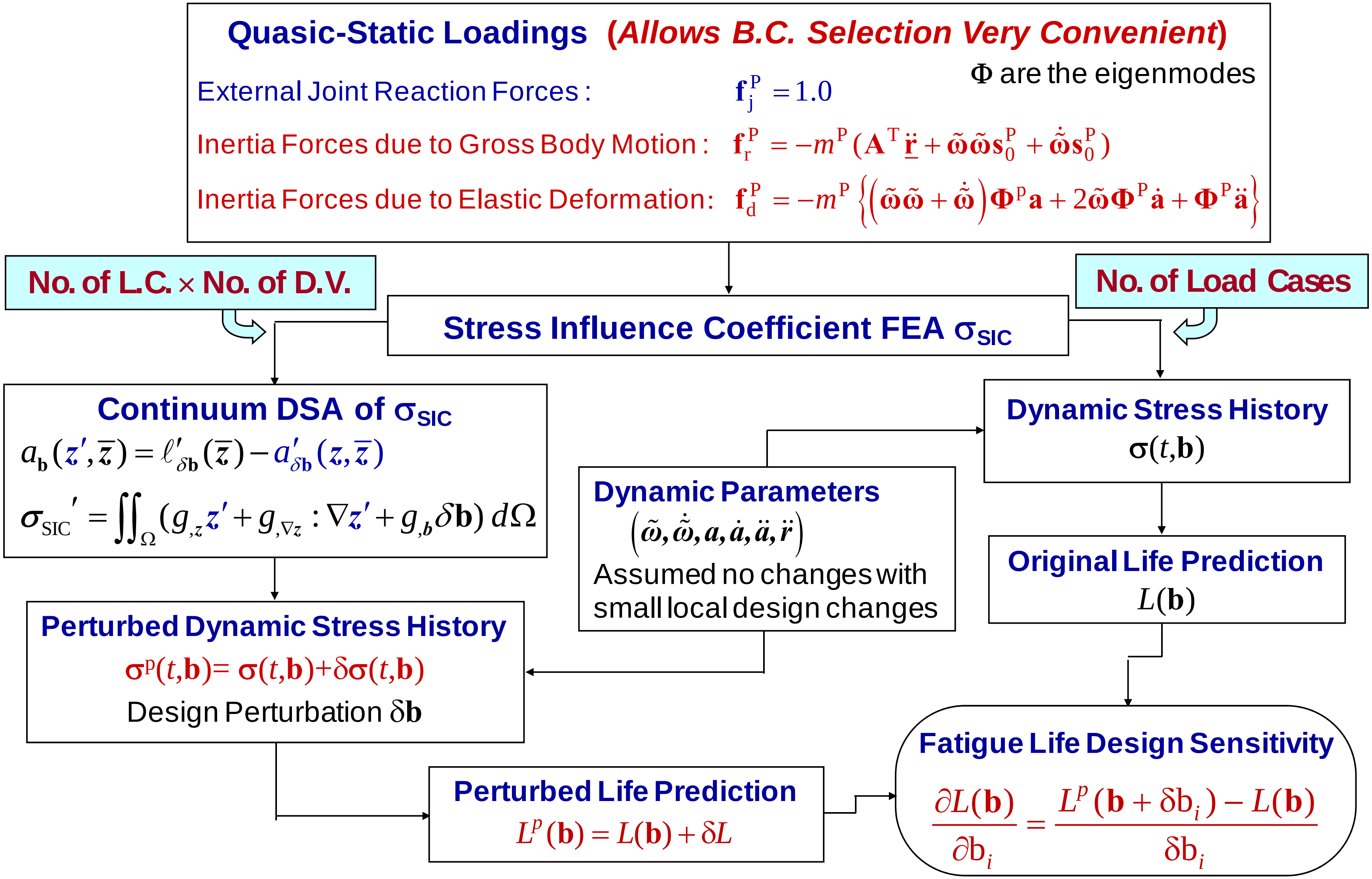
……

Evaluation of MPP candidate DSO, DRAW

Convergence Check

STOP

Computational Process in DRAW and DSO



Quasic-Static Loadings ***(Allows B.C. Selection Very Convenient)***

Φ are the eigenmodes

External Joint Reaction Forces : $\mathbf{f}_{\mathrm{j}}^{\mathrm{P}} = 1.0$

Inertia Forces due to Gross Body Motion : $\mathbf{f}_{\mathrm{r}}^{\mathrm{P}} = -m^{\mathrm{P}}(\mathbf{A}^{\mathrm{T}}\underline{\ddot{\mathbf{r}}} + \tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}}\mathbf{s}_0^{\mathrm{P}} + \dot{\tilde{\boldsymbol{\omega}}}\mathbf{s}_0^{\mathrm{P}})$

Inertia Forces due to Elastic Deformation: $\mathbf{f}_{\mathrm{d}}^{\mathrm{P}} = -m^{\mathrm{P}}\left\{\left(\tilde{\boldsymbol{\omega}}\tilde{\boldsymbol{\omega}} + \dot{\tilde{\boldsymbol{\omega}}}\right)\boldsymbol{\Phi}^{\mathrm{P}}\mathbf{a} + 2\tilde{\boldsymbol{\omega}}\boldsymbol{\Phi}^{\mathrm{P}}\dot{\mathbf{a}} + \boldsymbol{\Phi}^{\mathrm{P}}\ddot{\mathbf{a}}\right\}$

No. of L.C. × No. of D.V.

Stress Influence Coefficient FEA σ_{SIC}

No. of Load Cases

Continuum DSA of σ_{SIC}

$$a_{\mathbf{b}}(\mathbf{z}',\bar{\mathbf{z}}) = \ell'_{\delta\mathbf{b}}(\bar{\mathbf{z}}) - a'_{\delta\mathbf{b}}(\mathbf{z},\bar{\mathbf{z}})$$

$$\boldsymbol{\sigma}_{\mathrm{SIC}}{}' = \iint_{\Omega}(g_{,\mathbf{z}}\mathbf{z}' + g_{,\nabla\mathbf{z}} : \nabla\mathbf{z}' + g_{,\mathbf{b}}\delta\mathbf{b})\,d\Omega$$

Dynamic Stress History

$\boldsymbol{\sigma}(t,\mathbf{b})$

Dynamic Parameters

$\left(\tilde{\boldsymbol{\omega}}, \dot{\tilde{\boldsymbol{\omega}}}, \boldsymbol{a}, \dot{\boldsymbol{a}}, \ddot{\boldsymbol{a}}, \ddot{\boldsymbol{r}}\right)$

Assumed no changes with small local design changes

Original Life Prediction

$L(\mathbf{b})$

Perturbed Dynamic Stress History

$\boldsymbol{\sigma}^{\mathrm{P}}(t,\mathbf{b}) = \boldsymbol{\sigma}(t,\mathbf{b}) + \delta\boldsymbol{\sigma}(t,\mathbf{b})$

Design Perturbation $\delta\mathbf{b}$

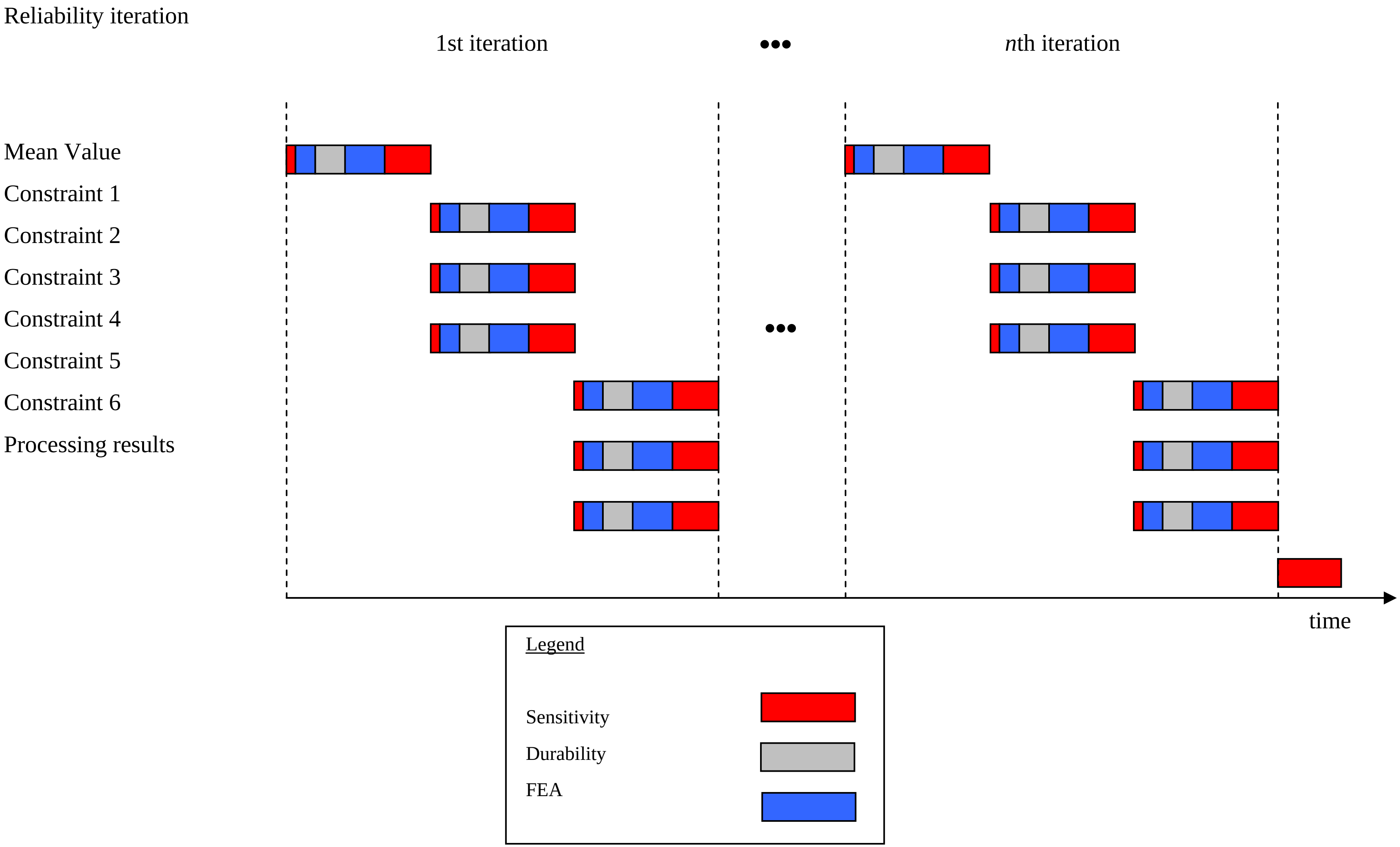
Perturbed Life Prediction

$L^{p}(\mathbf{b}) = L(\mathbf{b}) + \delta L$

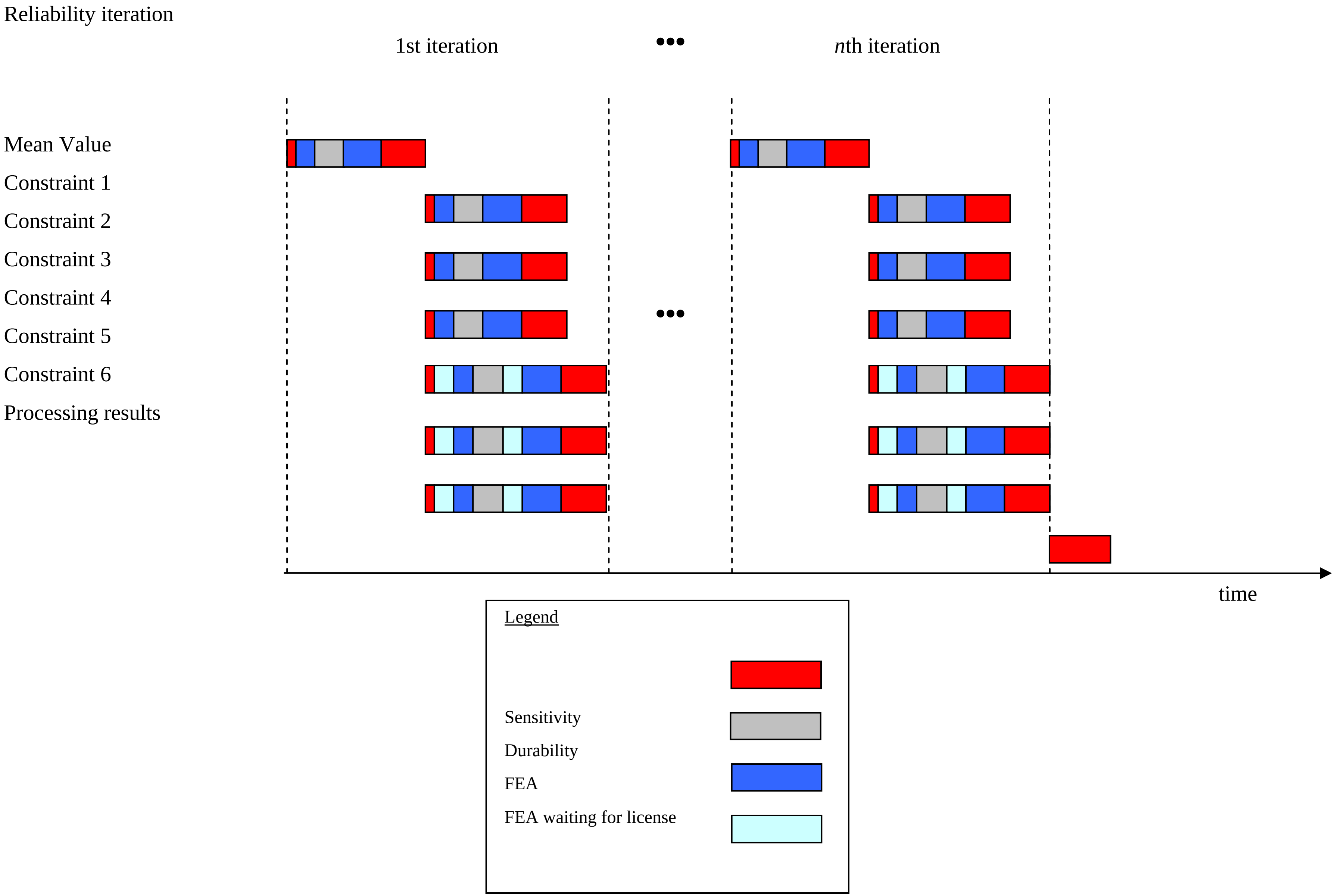
Fatigue Life Design Sensitivity

$$\frac{\partial L(\mathbf{b})}{\partial \mathrm{b}_i} = \frac{L^{p}(\mathbf{b} + \delta\mathrm{b}_i) - L(\mathbf{b})}{\delta\mathrm{b}_i}$$

GANTT Chart for 3 Licenses



Another GANTT Chart



Reliability iteration

1st iteration

•••

*n*th iteration

Mean Value

Constraint 1

Constraint 2

Constraint 3

Constraint 4

Constraint 5

Constraint 6

Processing results

•••

time

Legend

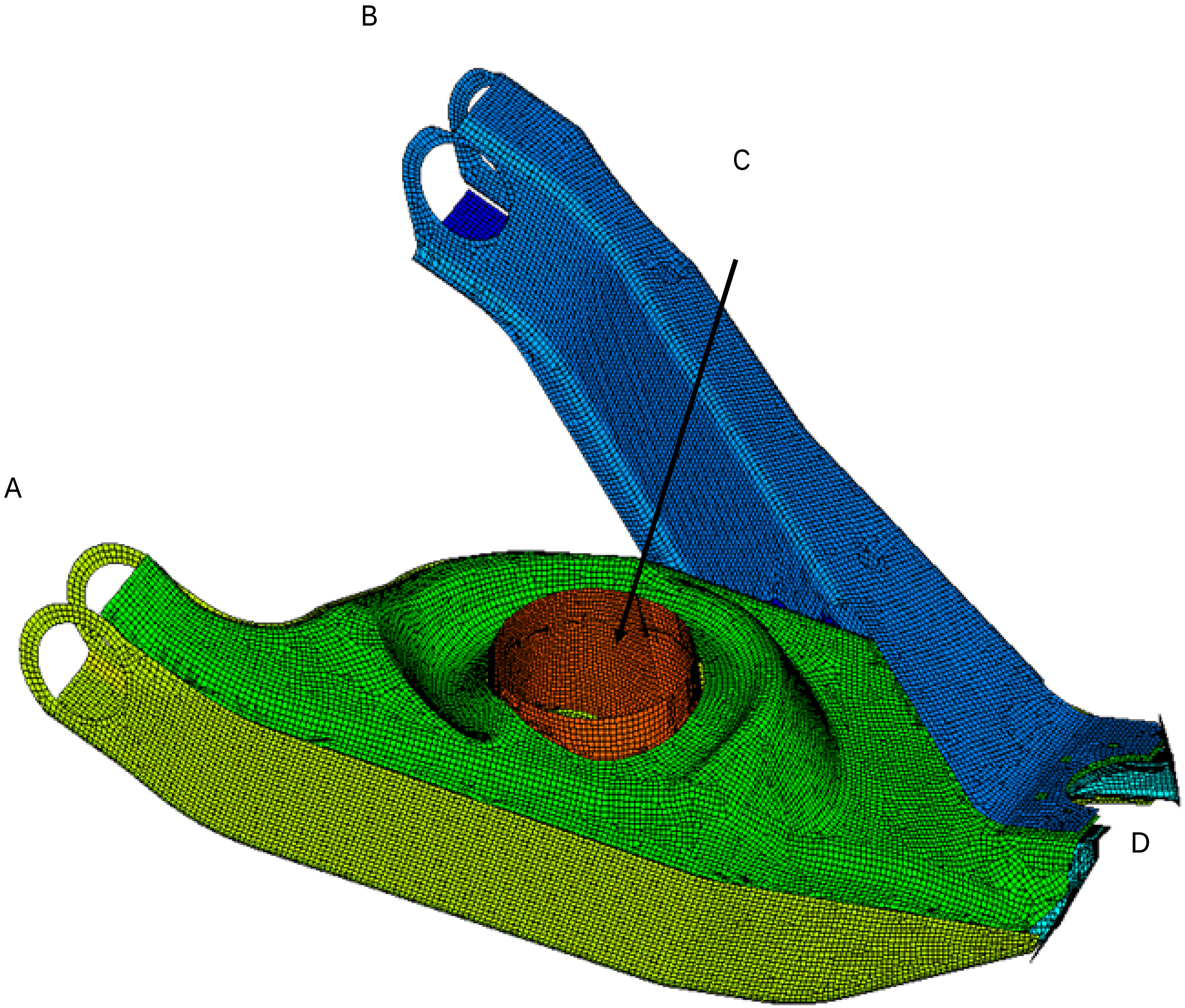
Sensitivity

Durability

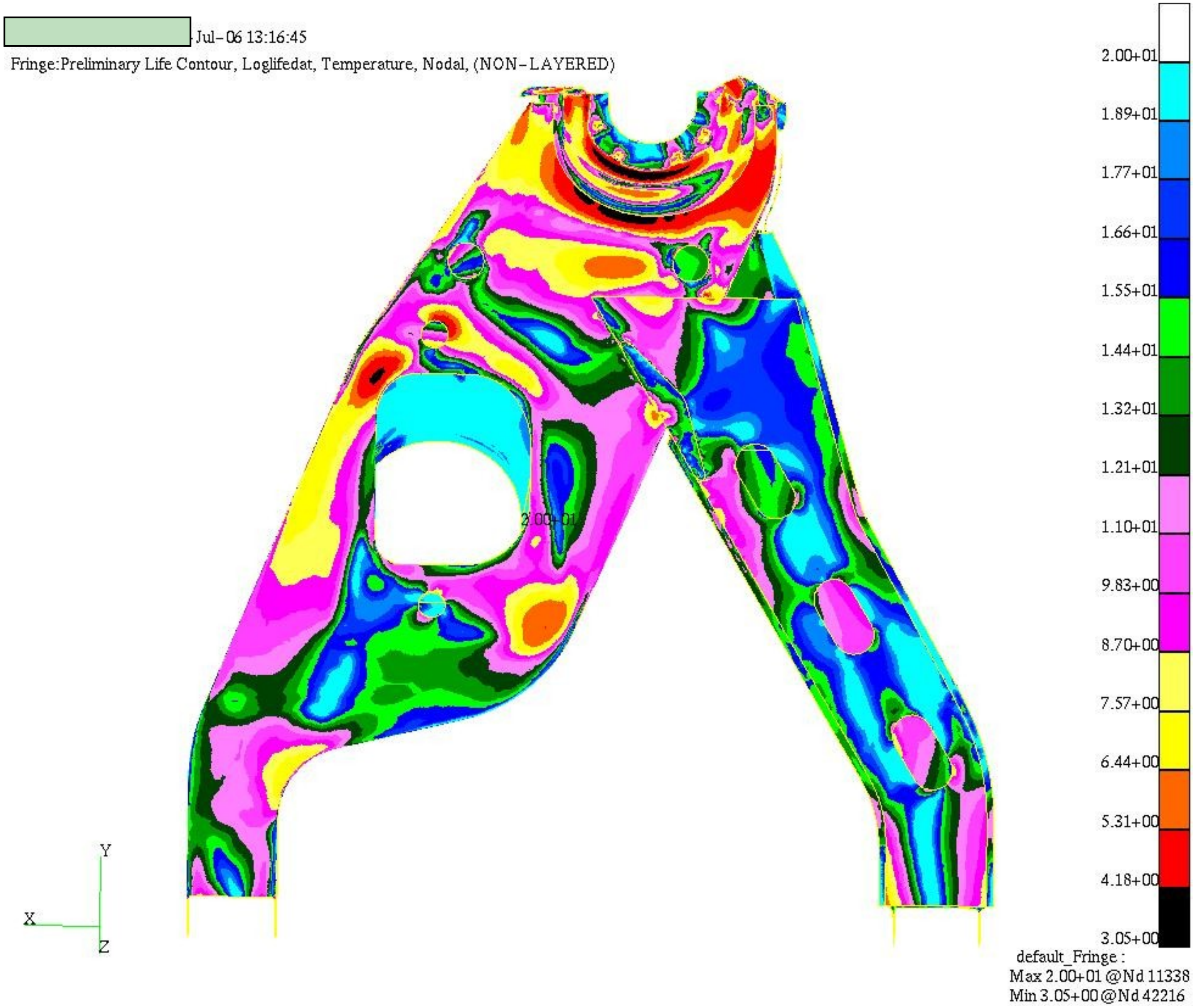
FEA

FEA waiting for license

The Lower A-Arm



A-Arm stress plot (initial)



Reliability-Based Design Optimization Results

- The cost (volume) increased from 111.4 in^3 to 136.9 in^3.
- Fatigue life increased from 5.31 x 10^4 to 1.0 x 10^6.
- The optimization converged in 4 design iterations.
- This required 100 function evaluations, and took 1397 minutes (23.3 hours) when run in serial mode (benchmark).
- With the 16 licenses of FE solver software and using parallel execution on 16 processors, took about 206 minutes (3 hours 26 minutes).
- This was a speed-up by a factor of 6.78 over serial processing.
- Some inefficiencies still existed in the code.

Scalability Results

	Run #	No of constr.	No of licenses	No of proc.	Ave. runtime (per constraint)	Ave. idle time (per processor)	Time (PR)
Training runs	**1**	**15**	**1**	**1**	**93.1**	**0.0**	**1397**
	2		**2**	**8**	**136.4**	**35.3 (282)**	**291**
	3		**4**	**8**	**125.1**	**23.6 (189)**	**259**
	4		**8**	**8**	**121.1**	**16.5 (132)**	**244**
	5		**2**	**15**	**179.1**	**57.6 (864)**	**237**
	6		**4**	**15**	**187.7**	**28.5 (428)**	**217**
	7		**8**	**15**	**191.8**	**13.6 (204)**	**206**
	8		***16***	***15***	***184.9***	***17.3 (259)***	***203***
	9	**30**	**1**	**1**	**94.1**	**0.0**	**2822**
	10		**2**	**8**	**126.5**	**53.8 (430)**	**529**
	11		**4**	**8**	**123.9**	**37.3 (298)**	**502**
	12		**8**	**8**	**122.4**	**32.3 (258)**	**492**
	13		**2**	**15**	**176.7**	**65.3 (979)**	**419**
	14		**4**	**15**	**170.9**	**33.2 (498)**	**376**
	15		**8**	**15**	**168.6**	**15.9 (239)**	**354**
	16		***16***	***15***	***165.7***	***14.0 (210)***	***346***
	17	**30**	**2**	**30**	**324.2**	**122.8 (3684)**	**448**
	18		**4**	**30**	**330.1**	**63.6 (1909)**	**395**
	19		**8**	**30**	**339.9**	**41.2 (1236)**	**382**
	20		**16**	**30**	**340.8**	**30.0 (901)**	**372**
Test runs	**21**	**15**	**7**	**10**	**125.7**	**53.2 (532)**	**242**
	22	**30**	**15**	**20**	**190.9**	**64.5 (1289)**	**352**

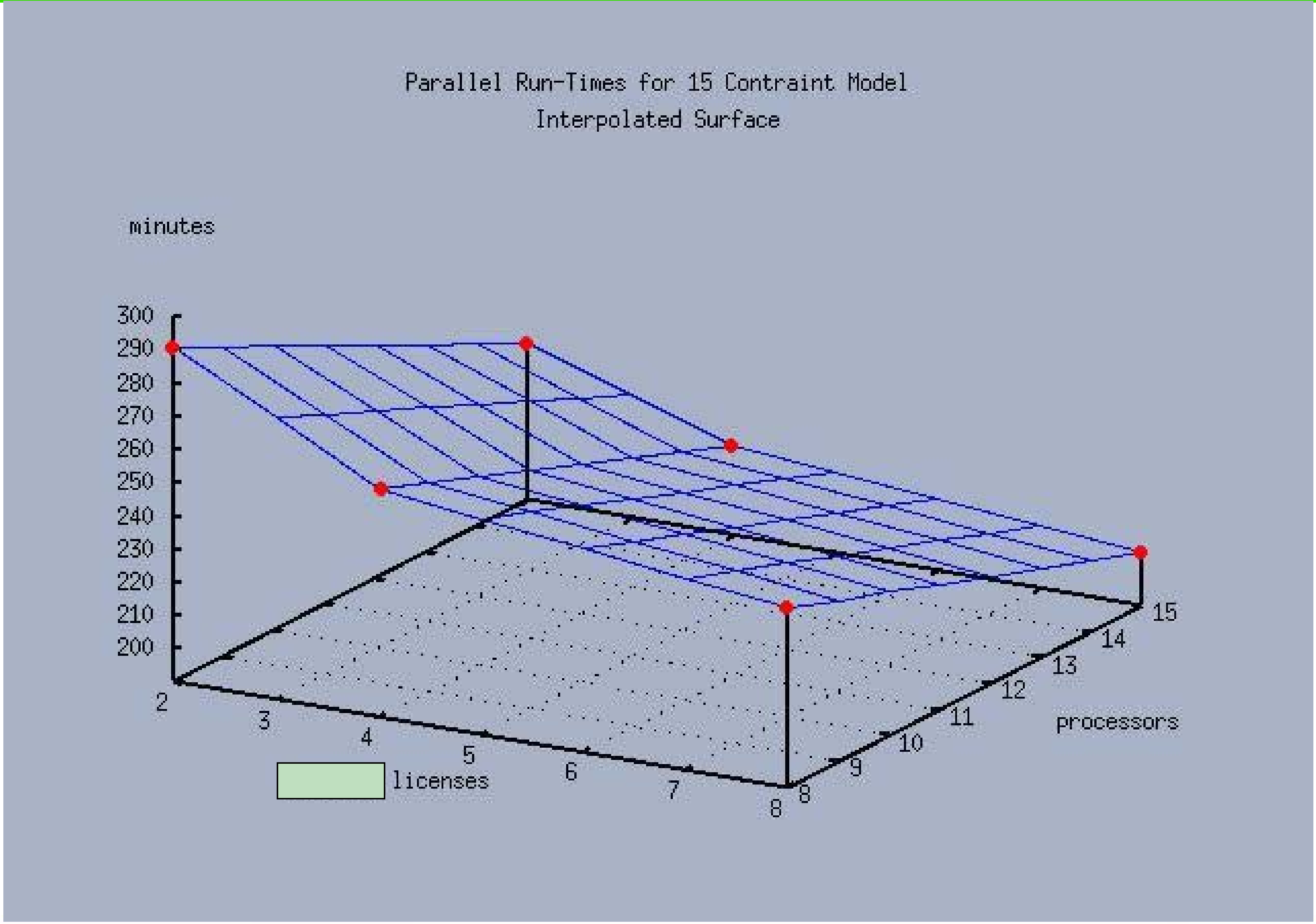
Legend for Defining Runtimes

- For:
- PR = parallel runtime in real time
- CR = total computational runtime, summed up over the processors
- I = total idle time, summed up over the processors
- np = number of processors
- nc = number of constraints
- we have:
- PR = (CR + I) / np
- or:
- PR = (CR / nc) * (nc / np) + I / np
- That is,
- parallel runtime in real time = (ave. computational runtime)*(ratio of constraints to processors) + ave. processor idle time

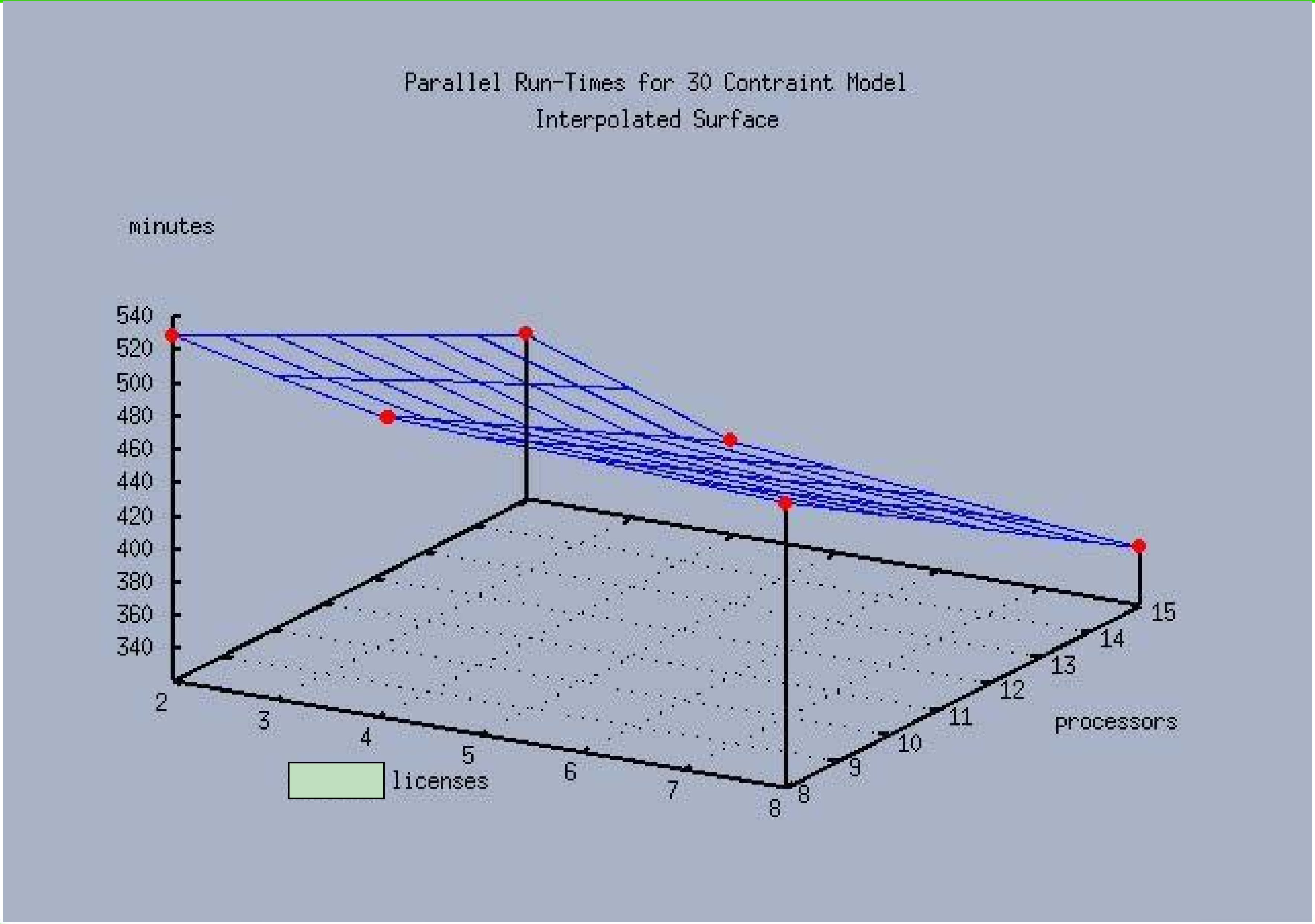
Trends observed in pilot runs

- The following trends appear significant from the data:
- Increasing the number of licenses decreases the average idle time per processor.
- Increasing the number of processors increases the average computational runtime per constraint.
- Increasing the number of licenses decreases the average runtime per constraint (when np<nc) and increases the average runtime per constraint (when np=nc.)
- Increasing the number of processors decreases the average idle time per processor if the number of licenses is 8 and increases the average idle time per processor if the number of licenses is 2, with no consistent trend when the number of licenses is 4.

Scalabiltiy Surface



Scalability Surface



Many Challenges

- Configuring number of licenses, processors, constraints
  - More processors than licenses?
  - One processor per constraint?
  - How does this scale?
- Memory and I/O problems
  - We had unexplained Scratch/Swap memory overutilization
  - I/O has been a constant issue
  - “Supercomputer – (definition) a devise for transforming a compute bound problem into an I/O bound problem”
- Cost of licenses
  - We must get better ‘package’ pricing for massively parallel runs from COTS software suppliers
  - Or, we must instead use “home-grown” code

Conclusions

- Follow-on project to start in August/September time frame
- More processors (over 100), More FEA licenses (32)
- Multi-component